Project 3 – Web Security
Part 2
CS155 – Indrajit “Indy” Khare

Outline

• Administrative
• Requirement Overview
• Attack A Defenses
• Attack B Defenses
• Attack C Defenses
• Attack D Defenses
• Extra Fun Defenses
• Other Notes
Administrative

- Due Monday June 1\textsuperscript{st}
- No more late days are allowed
- Setup cgi-bin on your su network account TODAY (linked from instructions)

Requirements

- Defend against all known attacks from Part 1
- Defend against all XSS an XSRF in zoobar.org (except login)
- Make sure you read non-goals section in assignment
  - Don’t add any new files
  - Don’t change DB
  - Don’t edit files in includes/
Attack A Defenses

• The attack is a simple XSS

• How do defend?
  – Do output sanitization

• From class:

  • PHP: `htmlspecialchars(string)`
    ```php
    & → &amp;   " → &quot;   ' → &apos;
    < → &lt;   > → &gt;
    ```

  – `htmlspecialchars("<a href=test>Test</a>", ENT_QUOTES);`

Outpus:
```html
&lt;a href=&amp;#039;test&amp;#039;&gt;Test&lt;/a&gt;
```

Attack B Defenses

• Simple XSRF (CSRF)

• How to Defend:
  – Secret Token
    • Ideally you use some HMAC with a secret
    • For this project you can simply hash the session token
    • Look at includes/auth.php for a lot of helpful code
Attack C Defenses

- Sniffing Login info
  - Secure the one non-html file that leaks the data
  - Modify it so that it doesn’t appear to do different things when logged in or not

- Phishing
  - Display warning if the user has visited a known bad page
  - Sniff browser history
    - Use make a hidden link to the bad url
    - Check generated link color via javascript
      ```javascript
      document.defaultView.getComputedStyle(document.getElementById("linkid"), "").getPropertyValue("color");
      ```
Attack D Defenses

- Don’t use eval!
- Make sure you are not displaying strings that can be bad

EF Defenses

- Go back and understand what the vulnerability is
  - Think quotes and event listeners
- Defense is very similar to Attack A
Hunting for Problems

• Look for wherever the website takes input

• Look for wherever the website outputs stuff that can be user generated

• Don’t worry about SQL Injection for this assignment

txt-db-api

• Third-party text file database library
• Data can be int, string, and autoincrement
• Need to escape strings: `' " \`
• Actually magic_quotes_gpc does this for us

```php
$recipient = $_POST['recipient']; // already escaped
$sql = "SELECT PersonID FROM Person WHERE Username='$recipient';
$rs = $db->executeQuery($sql);
if($rs->next())
    $id = $rs->getCurrentValueByName('PersonID);
```

Adapted from Collin Jackson 2007
PHP Sanitization Techniques

- **addslashes**(string)
  - Prepends backslash to ' " 
  - Already done by magic_quotes_gpc
  - Inverse: **stripslashes**(string)
- **htmlspecialchars**(string [, quote_style])
  - Converts & < > " to HTML entities
  - Use ENT_QUOTES to change ' to &apos;
- **strip_tags**(string [, allowable_tags])
  - Max tag length 1024
  - Does not sanitize tag properties
- **preg_replace**(pattern, replacement, subject)

More info: [http://php.net](http://php.net)

Adapted from Collin Jackson 2007

Questions?